Certo.

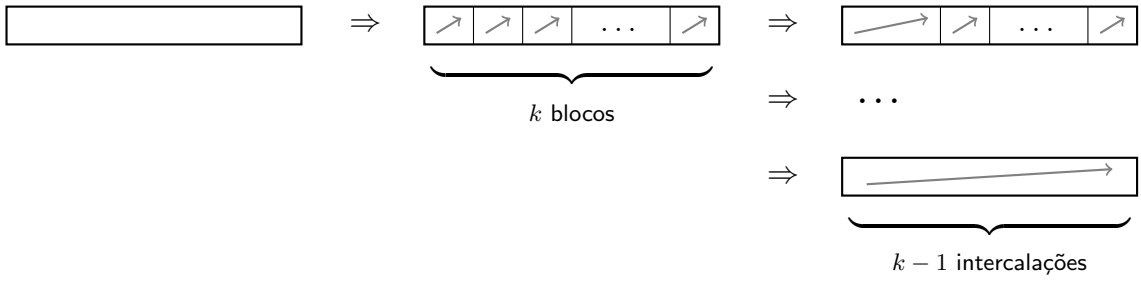
Agora já dá pra ver o que está acontecendo.

Quer dizer, quando a gente divide a lista em mais partes

- o tempo da ordenação diminui — (*o termo quadrático*)
- e o tempo da intercalação aumenta — (*o termo linear*)

Daí que, como o tempo da ordenação é maior, vale a pena continuar dividindo a lista em partes.

Então agora, nós consideramos o algoritmo que divide a lista em $k$ blocos




A coisa fica assim

```
ordenação-Bolha ( L[1..n/k] )
ordenação-Bolha ( L[n/k+1..2n/k] )
. . .
ordenação-Bolha ( L[(k-1)n/k+1..n] )

Intercalação ( L[1..n/k] , L[n/k+1..2n/k] )
. . .
Intercalação ( L[1..(k-1)n/k] , L[(k-1)n/k+1..n] )
```

— (*na prática, a gente faz isso com laços*)

E agora, é preciso ver quanto tempo isso leva.

**Análise**

Bom, a gente já sabe que cada ordenação pelo algoritmo da bolha vai levar tempo

$$O\Big( [\, n/k \,]^2 \Big) \quad = \quad O\Big( \frac{n^2}{k^2} \Big)$$

E como são k ordenações, isso nos dá um total de

$$k \;\times\; O\Big( \frac{n^2}{k^2} \Big) \quad = \quad O\Big( \frac{n^2}{k} \Big)$$

Daí, a gente pode imaginar que cada intercalação leva tempo $O(n)$.

E como são $k-1$ intercalações, isso nos dá um total de

$$(k-1) \;\cdot\; O(n)$$

Juntando as duas coisas, nós obtemos a seguinte estimativa para o tempo de execução do algoritmo

$$O\Big( \frac{n^2}{k} \Big) \quad + \quad (k-1) \cdot O(n)$$